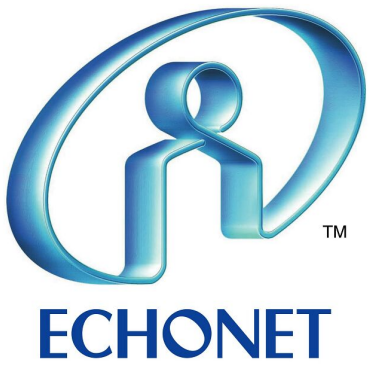
Personal Connected Health Alliance データ連携に関するガイドライン
ECHONET Lite Web API仕様例
Version 1.0.0



改定履歴

日付	版	説明
2023/4/28	Ver.1.0.0	制定、一般公開

目次

1. 本資料の位置づけ狙い

エコーネットコンソーシアムは、Personal Connected Health Alliance（以後"PCHA"と称す）と共同で「エコーネットコンソーシアム Personal Connected Health Alliance データ連携に関するガイダンス」Version 1.0（2022年4月14日）を策定した。両団体は快適で健康的な住まいの実現や、高齢化社会における介護負担などの社会課題解決に向けて、家電・住宅設備機器や健康機器から収集するデータを活用する仕組みの標準化を進めている。その活動を通して様々な関連サービスが普及する社会を目指している。

ガイダンスでは、データ連携イメージとして３つの構造案を掲載している。一般的にはサービス事業者側が主体となり、様々な事業者が提供するWeb APIを介してデータを集めてサービス化するデータ連携イメージ②、スーパーシティ構想などのデータ連携プラットフォームを介してオープンなデータ利活用を進めていくデータ連携イメージ③が想定されるが、上記に優先してデータ連携イメージ①としてECHONET Lite Web API対応サーバが、Personal Health Record（以後"PHR"と称す）データを扱う構造を掲載した。本構造はエコーネットコンソーシアム会員各社が健康データを積極的に収集し、主体的にサービス構築したり、サービス事業者への事業提案などを進めやすくするものでサービス化の普及促進を期待するものである。

特に、PHRデータの扱いについてはPCHA協力のもと、HL7 Fast Healthcare Interoperability Resource（以後"FHIR"と称す）サーバと連携することにより実現するモデルについて記述する。

本書の構成としては、第2章にてECHONET Lite Web APIサーバとFHIR®とのサーバ間連携モデルの全体構成案について示し、第3章にてFHIR®データ（PHRデータ）をECHONET Lite Web APIにて扱うための仕様案、第4章にてECHONET Lite Web APIサーバからFHIR®サーバへのアクセス例（認証・認可含む）について示す。

本書を活用し、PHRデータとIoT機器を連携させた新なスマートライフサービス（見守り・介護・健康住宅など）の構築や社会適用・貢献に繋げたい。

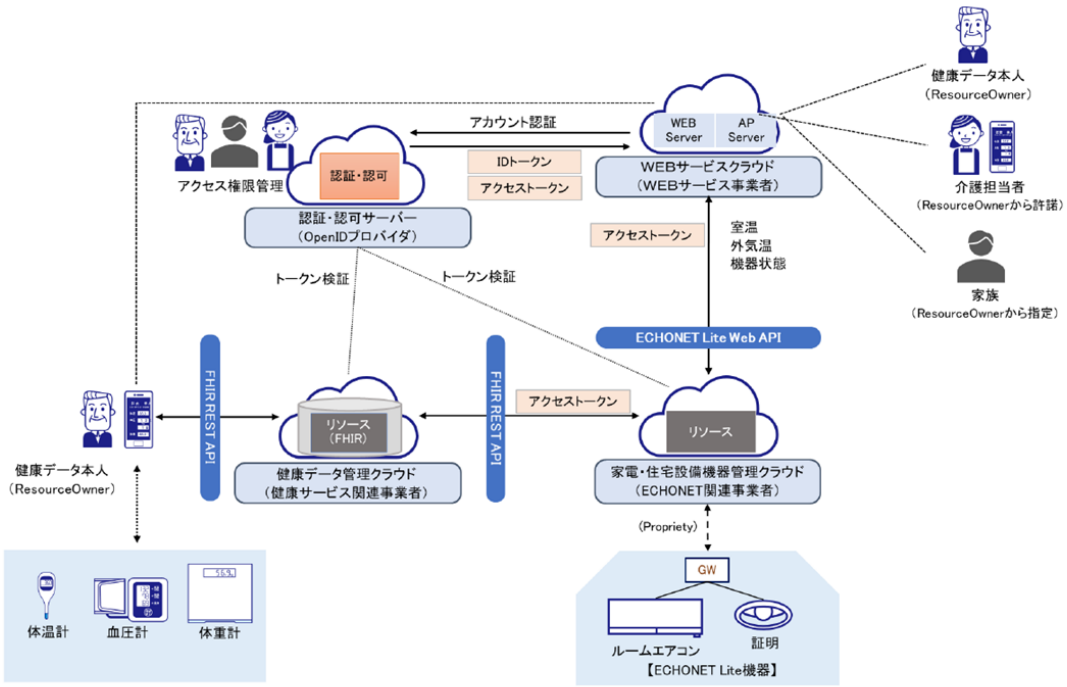
なお、本書では、「ECHONET Lite Web APIガイドライン（API仕様部）」をAPI設計におけるベースとしているため、対応サーバを構築する際には留意すること。また、機器仕様に関するWeb API仕様を合わせて使用する場合は、「ECHONET Lite Web APIガイドライン（機器仕様部）」を参照いただきたい。

注意点

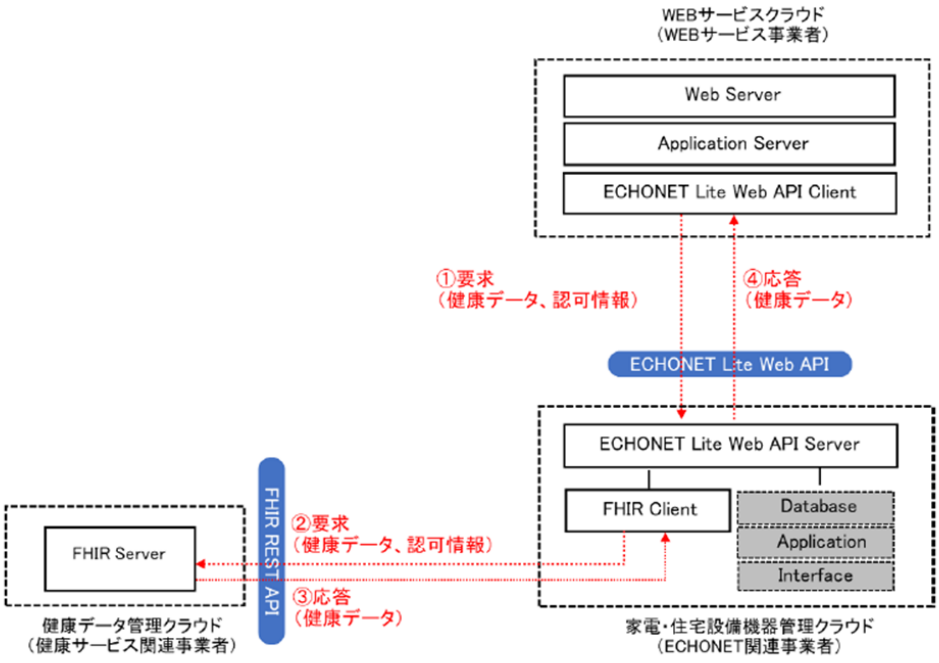
PHRについてはICT技術の発展によりデータの収集が容易化しており、それらのデータの利活用や標準化などについては活発に議論されている状況にあります。また、連携先であるFHIR®もリリース（R5）でPHRに関する扱いについて仕様拡張する方向で検討がされている。こうした状況を踏まえ、今回の検討では、対応するFHIRデータについては、人の情報（Patient）とFHIR®のバイタルサイン（一部）に限定することで、将来の標準化動向に応じて柔軟に改訂・仕様拡張することも視野に入れているに留意いただきたい。

2. サーバ間連携モデルの全体構成案

本書では、健康データと家電・住宅設備機器データを連携してサービス事業者に提供する方法として、健康データ、家電・住宅設備機器データをECHONET 関連事業者が合わせて、サービス事業者に提供するイメージを対象とする（「Personal Connected Health Alliance データ連携に関するガイドライン」の2.4 ①のイメージ）。



データ連携構造としては、下記のようにサービス事業者クラウド間インタフェースをECHONET Lite Web APIに一本化する（「Personal Connected Health Alliance データ連携に関するガイドライン」3.1 記載）。



3. FHIRデータのECHONET Lite Web API化

FHIRデータのECHONET Lite Web API化にあたり、基本的な設計思想は、「ECHONET Lite Web APIガイドライン（API仕様部）」に準拠する。

同API仕様部は、機器操作（監視・制御・通知）を中心とした記述となっているが、本FHIRデータを扱う場合にも、扱うリソースがdevicesから後述のhealthCareRecordsへ変更される点を除けば共通の仕様を多く含むため、適時参照いただき

たい。

なお、機器操作を実施したい場合は、同API仕様部と「ECHONET Lite Web APIガイドライン（機器仕様部）」とを合わせて参照すること。

3.1 healthCareRecords

PHRデータを扱うリソース種として "healthCareRecords" を導入する。 healthCareRecordsは、様々なヘルスケアデータを格納可能なリソースを想定しているが、 本書では、FHIRサーバにて扱うPHR相当のデータを対象に記述例を提示する。

3.1.1 ヘルスケアレコード一覧取得 GET /elapi/v1/healthCareRecords

ヘルスケアレコードIDの一覧を取得する。サーバに登録されたヘルスケアレコードIDが配列形式にて返却される。 登録が無い場合は、空の配列が返却される。

RESPONSE

STATUS CODE - 200

RESPONSE MODEL - application/json

Property	Type	Required	Description
healthCareRecords	array	Yes	---
healthCareRecords[].id	string	Yes	ヘルスケアレコードID
healthCareRecords[].descriptions	object	Yes	ヘルスケアレコードの名称。ヘルスケアレコードを特定するための名称。固有の名称が指定されることが望ましい。
healthCareRecords[].descriptions.ja	string	Yes	日本語記述
healthCareRecords[].descriptions.en	string	Yes	英語記述

【例】


```
{
  "healthCareRecords": [
    {
      "id": "12345",
      "descriptions": {
        "ja": "ユーザxx用",
        "en": "for user xx"
      }
    },
    {
      "id": "12346",
      "descriptions": {
        "ja": "ユーザyy用",
        "en": "for user yy"
      }
    }
```

```
  ]
}
```

【例】登録が無い場合

```
{
  "healthCareRecords": []
}
```

3.1.2 ヘルスケアレコードDescription取得 GET /elapi/v1/healthCareRecords/{healthCareRecordId}

ヘルスケアレコードIDを指定して、ヘルスケアレコードDescriptionを取得する。 ヘルスケアレコードDescriptionは、API仕様部記載のDevice Descriptionと同様、 ヘルスケアレコードに対応する仕様をJSON形式にて記述したものである。

REQUEST

PATH PARAMETERS

Property	Type	Required	Description
healthCareRecordId	string	Yes	ヘルスケアレコードID

【例】

```
"12345"
```

RESPONSE

STATUS CODE - 200

RESPONSE MODEL - application/json

Property	Type	Required	Description
healthCareRecordType		Yes	現状、PCHAにて扱うレコードを示す"pcha"のみ定義
descriptions	object	Yes	healthCareRecordTypeに関する記述
descriptions.ja	string	Yes	日本語記述
descriptions.en	string	Yes	英語記述
properties	object	Yes	-
properties.id	string	Yes	patient ID
properties.age	number	Yes	年齢
properties.gender	string	Yes	性別

Property	Type	Required	Description
properties.active	boolean	Yes	Patientレコードがアクティブ使用中
properties.bmi	number	Yes	BMI値
properties.bodyTemp	number	Yes	体温
properties.bodyWeight	number	Yes	体重
properties.bloodPressure	number	Yes	血圧
properties.heartRate	number	Yes	心拍数
properties.pulseOximetry	number	Yes	パルスオキシメータ（経皮的酸素飽和度）値

* 詳細については、3.1.3 にて記述。

【例】

後述の「Appendix 1. ヘルスケアレコードdescription例」にて記載。

3.1.3 ヘルスケアレコード内容取得 GET /elapi/v1/healthCareRecords/{healthCareRecordId}/properties

ヘルスケアレコードIDを指定して、GET対象となるすべての値を取得する。

REQUEST

PATH PARAMETERS

Property	Type	Required	Description
healthCareRecordId	string	Yes	ヘルスケアレコードID

【例】

```
"12345"
```

RESPONSE

STATUS CODE - 200

RESPONSE MODEL - application/json

Property	Type	Required	Description
id	string	Yes	patient ID
age	number	Yes	年齢。0もしくは正の整数。FHIR連携時はPatientのbirthDateから算出

Property	Type	Required	Description
gender	string	Yes	性別。"male", "female", "other", "unknown"のいずれか
active	boolean	Yes	Patientレコードがアクティブ使用中
bmi	number	Yes	BMI値。単位：kg/m^2 *
bodyTemp	number	Yes	体温。単位：℃ *
bodyWeight	number	Yes	体重。単位：kg *
bloodPressure	array[number]	Yes	拡張時、収縮時を配列で記載。単位：mmHg *
heartRate	number	Yes	心拍数。単位：/min *
pulseOximetry	number	Yes	パルスオキシメータ（経皮的酸素飽和度）値。単位：% *

* なお、小数点以下の桁数は、任意。

【例】

```
{
  "id": "0001",
  "age": 68,
  "gender": "male",
  "active": true,
  "bmi": 22.5,
  "bodyTemp": 36.7,
  "bodyWeight": 51.2,
  "bloodPressure": [90, 140],
  "heartRate": 61,
  "pulseOximetry": 97
}
```

4. ApplicationサーバからECHONET Lite Web APIサーバおよびECHONET Lite Web APIサーバからFHIRサーバへのアクセス例

「2. **サーバ間連携モデルの全体構成案**」に記載した構成案において、ApplicationサーバがECHONET Lite Web APIサーバ上の家電・住設機器データにアクセスするケースと、ApplicationサーバがECHONET Lite Web APIサーバ上のhealthCareRecordsリソースにアクセスするケースの、2通りのケースが存在する。ECHONET Lite Web APIサーバ上のhealthCareRecordsリソースに対するアクセスの場合、ECHONET Lite Web APIサーバがhealthCareRecordsリソースを提供するために、バックエンド側で連携先となるFHIRサーバとの間で必要なデータのやりとりを行う必要がある。

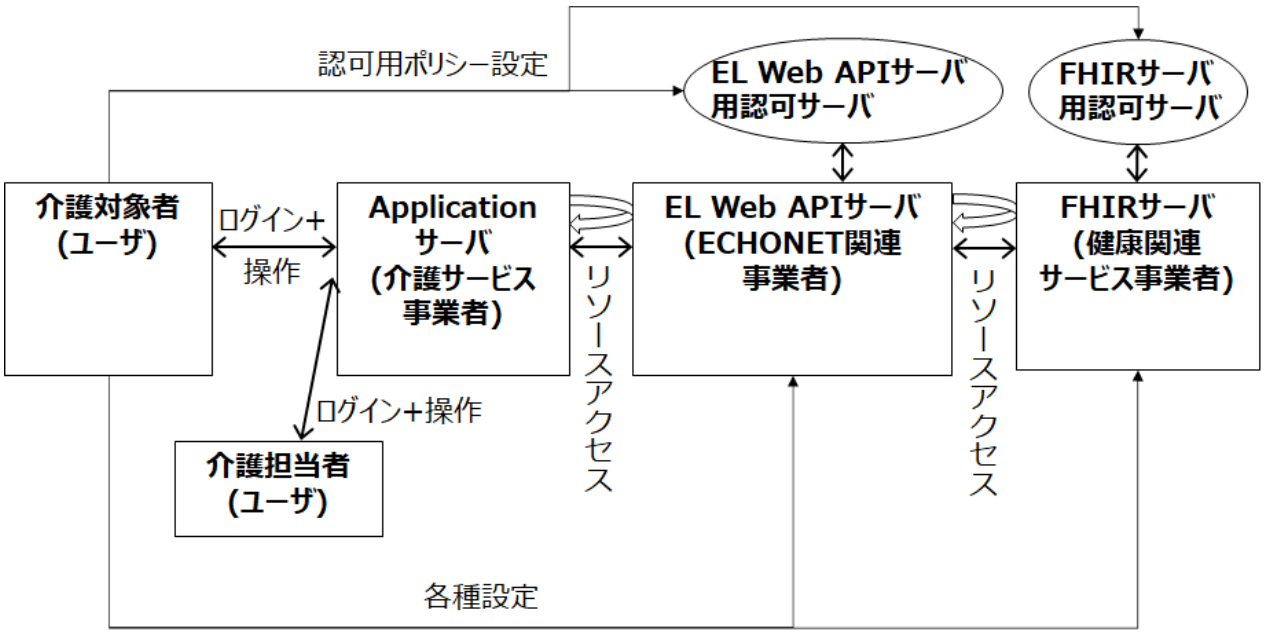
本節では、ApplicationサーバとECHONET Lite Web APIサーバ間、ECHONET Lite Web APIサーバとFHIRサーバ間のそれぞれでデータをやり取りする際に行う認可に関する前提条件と要件の例を説明する。また参考情報として、サーバ間の認可手順の例を記載する。

4.1 本節で想定するシステム構成例

本節で想定するシステム構成例を図に示す。図に示されるシステム構成要素の概要を以下に説明する。

- 介護対象者、介護担当者(ユーザ)
  - ブラウザ等をUIにして、Applicationサーバ上で操作を行う。
  - 介護対象者(リソース所有者)は、ECHONET Lite Web APIサーバとFHIRサーバ(いずれもリソースサーバ)、および、ECHONET Lite Web APIサーバ用認可サーバとFHIRサーバ用認可サーバに対して、サーバ間連携におけるリソースアクセスに必要な設定を行う。
- Applicationサーバ(介護サービス事業者)
  - ユーザの操作をトリガに、ECHONET Lite Web APIサーバ上のリソースにアクセスする。
- ECHONET Lite Web APIサーバ(ECHONET関連事業者)
  - Applicationサーバがアクセスするリソースを管理する。
  - Applicationサーバがアクセスしたリソースを返すためにFHIRサーバ上のリソースにアクセスが必要な場合は、FHIRサーバにアクセスしてApplicationサーバへ返す。
- ECHONET Lite Web APIサーバ用認可サーバ
  - ECHONET Lite Web APIサーバ上のリソースへのアクセスに対する認可判断と、アクセストークン発行を行う。
- FHIRサーバ(健康関連サービス事業者)
  - ECHONET Lite Web APIサーバを介してApplicationサーバがアクセスするリソースを管理する。
- FHIRサーバ用認可サーバ
  - FHIRサーバ上のリソースへのアクセスに対する認可判断と、アクセストークン発行を行う。

リソースサーバ上のリソースへのアクセスに対する認可判断は、各リソースサーバ用の認可サーバが発行し、リソースアクセス時にアクセス元がリソースサーバに提示するアクセストークンに基づき行うことを想定する。



4.2 前提条件

本節では、具体的な手順は本文書に記載しないが、サーバ間連携を実現するために必要な項目を前提条件として記載する。

- ApplicationサーバとECHONET Lite Web APIサーバの間で、契約などで事前に信頼関係を確立している。
- ECHONET Lite Web APIサーバとFHIRサーバの間で、ECHONET Lite Web APIサーバがFHIRサーバから取得したデータをApplicationサーバに提供する場合があることも含めて、契約などで事前に信頼関係を確立している。

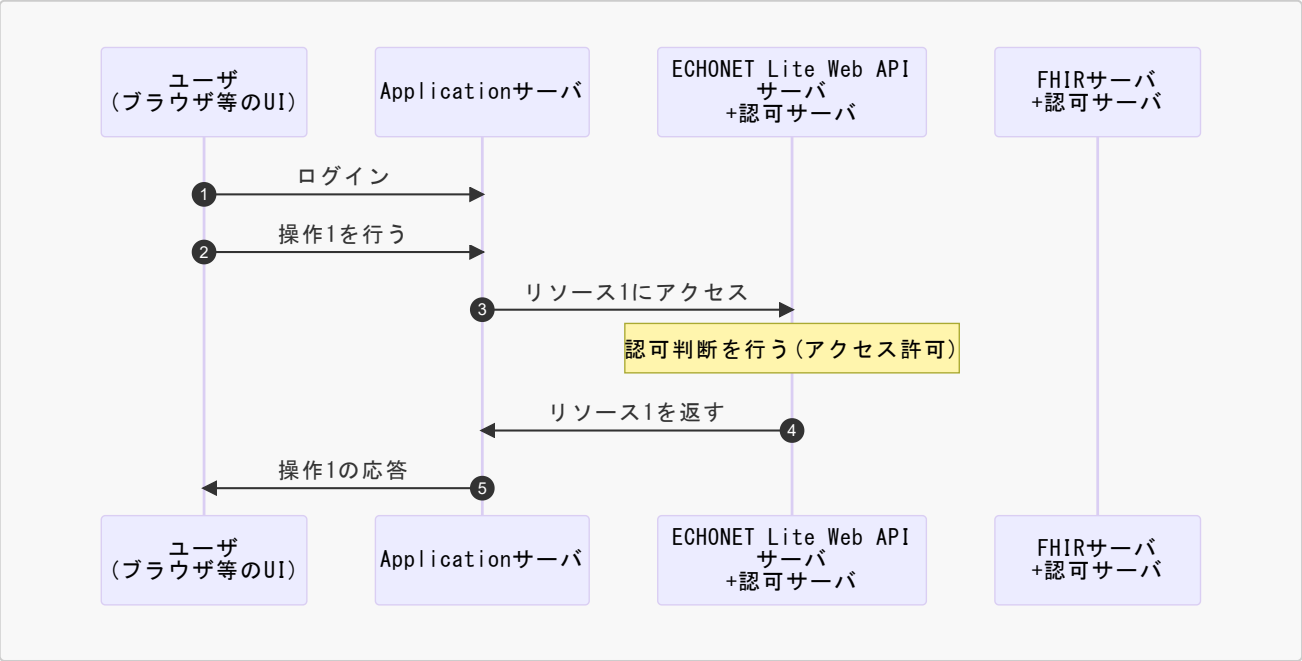
- 介護対象者(リソース所有者)は、各リソースサーバと認可サーバに、事前にサーバ間連携におけるリソースアクセスに必要な設定を行うことができる。
- アクセス元サーバがアクセス先サーバに対してアクセス対象リソースを指定するために必要な、アクセス先サーバにおけるID情報(例 ApplicationサーバがECHONET Lite Web APIサーバにアクセスする際のdeviceID、ECHONET Lite Web APIサーバがFHIRサーバにアクセスする際のpatientID)を、アクセス元サーバが知るための方法が提供される。

4.3 サーバ間連携の手順概要

本節では、サーバ間連携の手順概要を説明する。

ApplicationサーバがECHONET Lite Web APIサーバ上の家電・住設機器データ(リソース)にアクセスするケース

1. ユーザがApplicationサーバにログインする。
2. ユーザが操作1を行う。
3. Applicationサーバは、操作1をトリガにECHONET Lite Web APIサーバ上の家電・住設機器データ(リソース1)にアクセスする。
4. ECHONET Lite Web APIサーバ用認可サーバでリソース1に対する認可判断を行い、アクセスを許可する場合はApplicationサーバにリソース1を返す。
5. Applicationサーバはユーザに操作1の応答を返す。

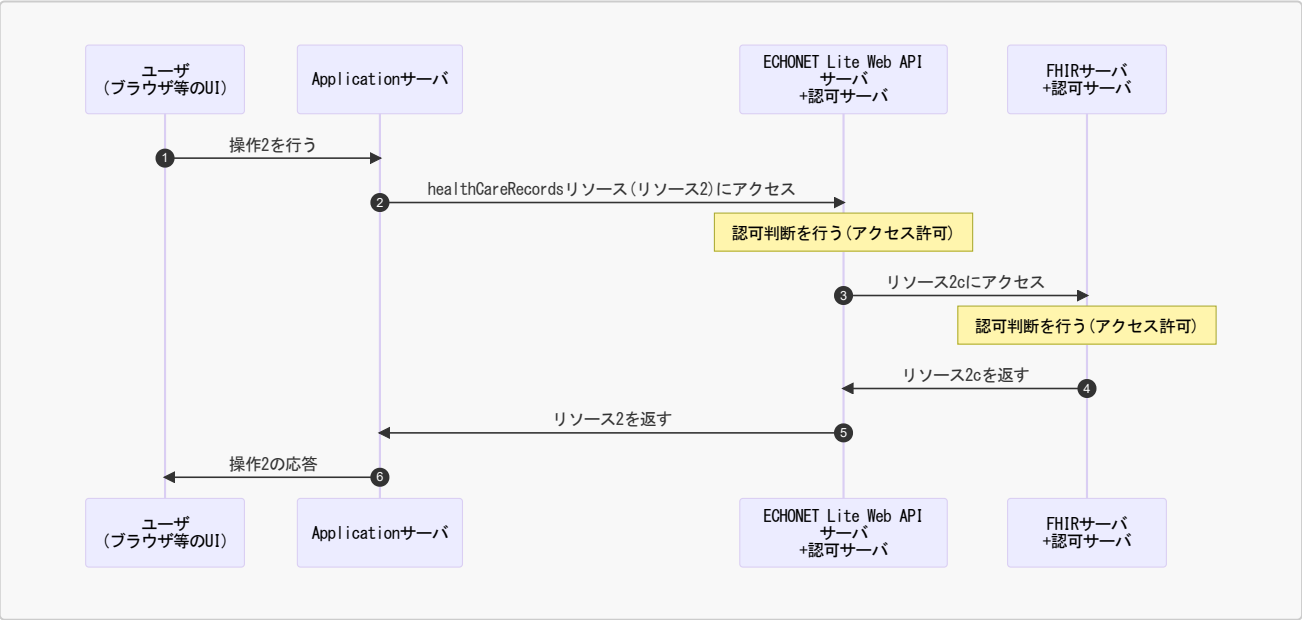


ApplicationサーバがECHONET Lite Web APIサーバ上のhealthCareRecordsリソースにアクセスするケース

以下の手順では、ユーザはApplicationサーバにログイン済みであることを仮定する。

1. ユーザが操作2を行う。
2. Applicationサーバは、操作2をトリガにECHONET Lite Web APIサーバ上のhealthCareRecordsリソース(リソース2)にアクセスする。
3. ECHONET Lite Web APIサーバ用認可サーバでリソース2に対するアクセス認可判断を行った上で、ECHONET Lite Web APIサーバがFHIRサーバのリソース2cにアクセスする。

4. FHIRサーバはFHIRサーバ用認可サーバで認可判断を行い、アクセスを許可する場合はECHONET Lite Web APIサーバにリソース2cを返す。
5. FHIRサーバからリソース2cを返されると、 ECHONET Lite Web APIサーバはApplicationサーバにリソース2を返す。
6. Applicationサーバはユーザに操作2の応答を返す。



4.4 サーバ間連携における認可手順に求められる要件

本節では、PCHAデータ連携を実現する際のサーバ間連携における認可手順に求められる要件を記載する。

1. ユーザは、自身が所有者ではないリソースにアクセス可能とする。

   介護担当者(=ユーザ)が介護対象者(=リソース所有者)のデータにアクセスする必要があるため、本要件を挙げた。

2. リソースサーバ用の認可サーバは、ユーザ，ユーザの所属組織，アクセス元サーバなどに基づく認可判断を行うことを可能とする。

   PCHA連携では以下のような複数の考え方が想定され得るため、本要件を挙げた。

- 認可サーバが、ユーザに基づく認可判断を行う。

  例: 介護担当者は自分が担当する介護対象者のデータのみアクセス可能とするため、ユーザに基づく認可判断を行う。

- 認可サーバが、ユーザの所属組織(例 介護サービス事業者)に基づく認可判断を行う。

  例: 介護担当者は介護サービス事業者の顧客である介護対象者のデータのみアクセス可能とするため、ユーザの所属組織に基づく認可判断を行う。

- 認可サーバは、アクセス元サーバに基づく認可判断を行う。

  例: FHIRサーバは、ECHONET Lite Web APIサーバがアクセス元である場合にアクセスを許可するリソースを契約に基づき決定する。FHIRサーバ上のリソースに対するアクセスにユーザやユーザの所属組織に基づく認可判断が必要な場合、サーバ間の信頼関係に基づき、FHIRサーバはECHONET Lite Web APIサーバが行う認可判断に任せる。

3. リソースアクセスに対する認可判断は、リソース所有者が事前に設定した条件(ポリシー)に基づきリソース所有者の介在なく行う方法と、リソースアクセス発生時に都度リソース所有者の承認を得る方法を、リソース毎に選択可能とすることが望ましい。

   介護担当者(=ユーザ)が介護対象者(=リソース所有者)の傍にいる状況で介護対象者のデータにアクセスするとは限らないので、システムの操作性を良くするためリソース所有者の介在なく行う方法を要件に挙げた。 リソースの中に要配慮個人情報に分類されるデータが含まれる場合などを考慮し、リソースアクセス発生時に都度リソース所有者の承認を得る方法を要件に挙げた。

4. FHIRサーバ用認可サーバは、リソースアクセスの大本の要求元(Applicationサーバ)からECHONET Lite Web APIサーバを介してリソースアクセス要求を受け取る場合でも、ユーザやユーザの所属組織に基づく認可判断を行うことが可能であることが望ましい。

   ApplicationサーバがECHONET Lite Web APIサーバ経由でFHIRサーバ上のリソースにアクセスするケースで、FHIRサーバ用認可サーバがユーザやユーザの所属組織に基づく認可判断を行うケースに対応可能とするため、本要件を挙げた。

4.5 サーバ間連携における認可手順の例(参考情報)

本節では、ユーザがApplicationサーバにログイン済みの状態で行った操作をトリガとして、Applicationサーバが ECHONET Lite Web APIサーバ上の家電・住設機器データにアクセスするケースと、ApplicationサーバがECHONET Lite Web APIサーバ上のhealthCareRecordsリソースにアクセスしてECHONET Lite Web APIサーバ経由でFHIRサーバが管理するリソースにアクセスするケースの認可手順例を説明する。参考情報として記載しており、本節に記載する手順例以外の手順の使用を妨げるものではない。

下記の手順は以下のことを想定している。

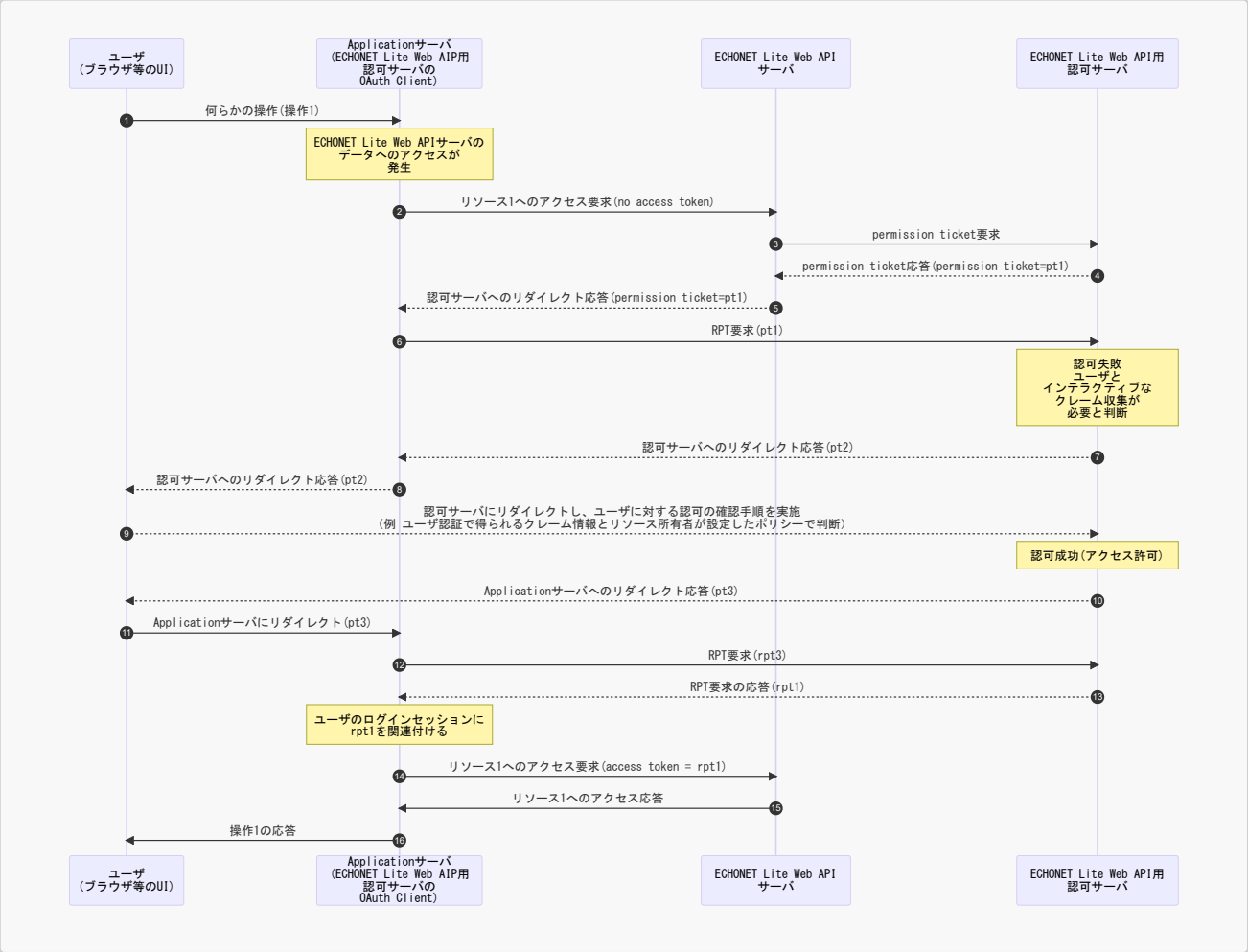
- ECHONET Lite Web APIサーバ用認可サーバは、ユーザに基づく認可判断を行う。
- FHIRサーバ用認可サーバは、ECHONET Lite Web APIサーバに対してサーバに基づく認可判断を行う。
- FHIRサーバ上のリソースに対応付けられるECHONET Lite Web APIサーバ上のリソース (例えばhealthCareRecord)へのアクセスに対して、ECHONET Lite Web API用認可サーバがユーザに基づく認可判断を行う。これにより、ApplicationサーバによるFHIRサーバ上のリソースへの(間接的)アクセスに対して、システムとしてはユーザに基づく認可判断を行うことを可能とする。

以下の手順例は、UMA 2.0 (https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-grant-2.0.html)に基づいている。

ApplicationサーバがECHONET Lite Web APIサーバ上の家電・住設機器データにアクセスするケース

1. ユーザが何らかの操作(操作1)を行う。
2. Applicationサーバが、ECHONET Lite Web APIサーバのリソース(リソース1)へのアクセス要求(no access token)を送信する。
3. permission ticket要求を送信する。
4. permission ticket応答(pt1)を送信する。
5. 認可サーバへのリダイレクト応答(pt1)を送信する。
6. RPT (Requesting Party Token、access tokenの一種)要求(pt1)を送信する。
7. 認可サーバへのリダイレクト応答(error: need_info, new permission ticket: pt2)を送信する。
8. 認可サーバへのリダイレクト応答(permission ticket: pt2)を送信する。

9. リダイレクトし、ユーザに対する認可判断を実施(例 ユーザ認証で得られるクレーム情報と、リソース所有者が設定したポリシーで判断)する。
10. 認可判断の結果アクセスが許可され、Applicationサーバ(=Client)へのリダイレクト応答(permission ticket: pt3)を送信する。
11. Applicationサーバ(=Client)にリダイレクト(permission ticket: pt3)する。
12. RPT要求(pt3)を送信する。
13. RPT要求の応答(rpt1)を送信する。
14. ユーザのログインセッションにrpt1を関連付け、リソース1へのアクセス要求(access token=rpt1)を送信する。
15. リソース1へのアクセス応答を送信する。
16. 操作1の応答を送信する。



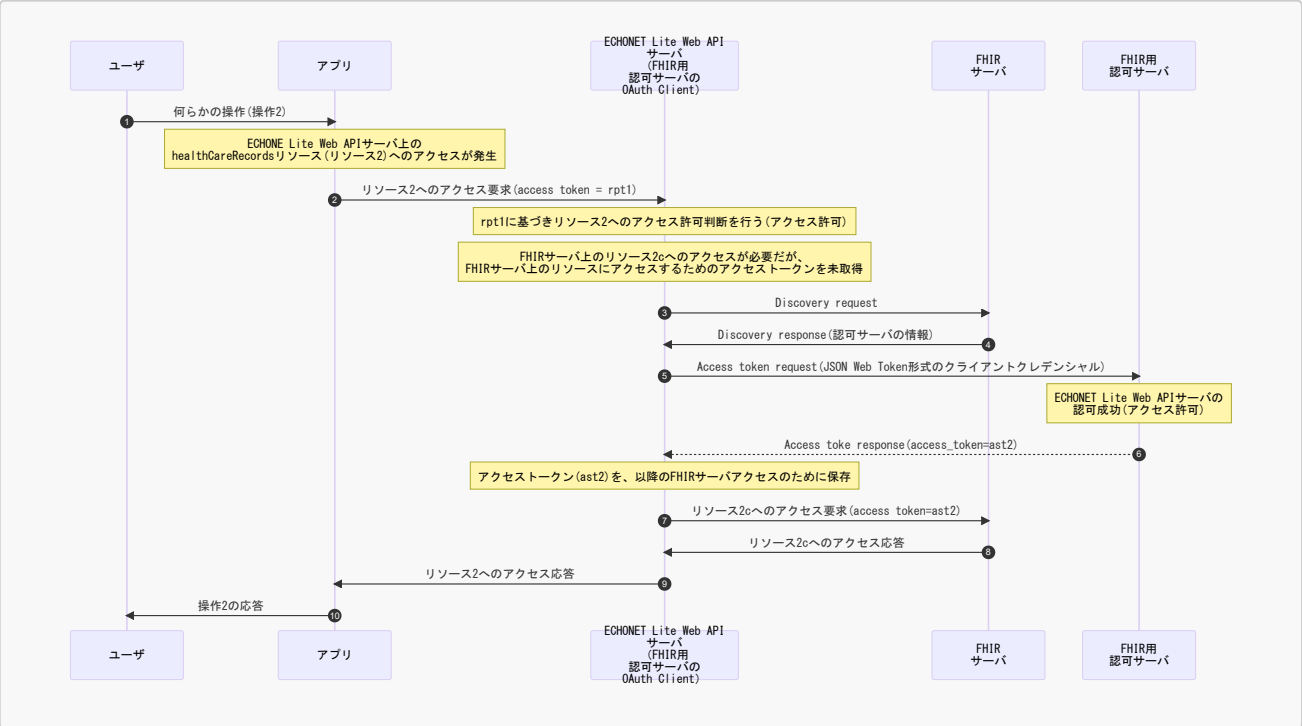
ApplicationサーバがECHONET Lite Web APIサーバ上のhealthCareRecordsリソースにアクセスするケース

Applicationサーバが上記手順を実行し、rpt1を取得済みの状態でアクセスする場合の例を以下に示す。

1. ユーザが何らかの操作(操作2)を実行する。
2. Applicationサーバが、ECHONET Lite Web APIサーバ上のhealthCareRecordsリソース(リソース2)へのアクセス要求(access token=rpt1)を送信する。
3. ECHONET Lite Webサーバは、rpt1に基づきリソース2へのアクセス許可判断を行う。アクセスを許可する場合、FHIRサーバ上のリソース2cにアクセスする必要があるが、FHIRサーバ上のリソースにアクセスするためのアクセストークンを未取得であるため、FHIRのSMART Backend Service (http://www.hl7.org/fhir/smart-app-

launch/backend-services.html）の手順を使いアクセストークンの取得を開始する。 FHIRサーバにDiscovery requestを送信する。

4. Discovery response(認可サーバの情報)を送信する。
5. OAuth2.0のClient Credentials flowに従いAccess token request(JSON Web Token形式のクライアントクレデンシャル)を送信する。
6. FHIRサーバ用認可サーバによるECHONET Lite Web APIサーバの認可判断が成功した場合(アクセス許可)、Access toke response(access_token=ast2)を送信する。
7. ECHONET Lite Web APIサーバはアクセストークンast2を記録し、リソース2cへのアクセス要求(access token=ast2)を送信する。
8. リソース2cへのアクセス応答を送信する。
9. リソース2へのアクセス応答を送信する。
10. 操作2の応答を送信する。



4.6 FHIRデータのアクセスと機器操作の連携

これまでFHIRデータへのアクセスをECHONET Lite Web APIサーバ経由で実施するための認証・認可の仕組みについて説明してきた。同FHIRデータへのアクセスが可能なユーザ（介護対象者、家族、介護担当者）は、機器（群）操作も同様に可能となる。 なお、機器操作のみを実施したい場合では、4.5前半の認証・認可の処理だけを実施すれば良く、後半の処理については不要となる。

ユーザが扱えるFHIRデータや操作可能な機器（群）をどのように連携させるかについては、ユーザが使用しているアプリの役割となる。

例えば、ヘルスケアデータ（体温）を下記にて取得。

```
リクエスト：
GET /elapi/v1/healthCareRecords/{healthCareRecordId}/properties/bodyTemp
```

```
レスポンス：
{ "bodyTemp": 37.5  }
```

次に、エアコンの動作状態、運転モード設定、温度設定値、室内温度計測値を取得。

```
リクエスト：
GET /elapi/v1/devices/{deviceId}/properties

レスポンス：
{
   "operationStatus": true,
   "operationMode": "cooling",
   "targetTemperature": 28,
   ”roomTemperatur": 28,
   ...
}
```

エアコンの設定温度を変更といった操作を一連の操作として提供するアプリが考えられる。

```
リクエスト：
PUT /elapi/v1/healthCareRecords/{healthCareRecordId}/properties/targetTemperature
{  "targetTemperature": 26  }

レスポンス：
{  "targetTemperature": 26  }
```

上記のアプリを実現するには、ユーザのアクセス対象となるhealthCareRecordIdやdeviceIdを上記API呼び出し前に入手しておく必要がある。初期状態のように、各IDが入手できていない場合、各々のリソースのID一覧取得APIを用いて各ID集合を入手し、その中から対象IDを選択するなどの操作も必要かもしれない。

Appendix

1. ヘルスケアレコードdescription例

```
{
    "healthCareRecordType": "pcha",
    "descriptions": {
        "ja": "PCHA連携用ヘルスケアレコード",
        "en": "healthcare record for PCHA linkage"
    },
    "properties": {
        "id": {
            "descriptions": {
                "ja": "patient ID",
                "en": "patient ID"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "string"
            }
```

```
        },
        "age": {
            "descriptions": {
                "ja": "年齢",
                "en": "age"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "number"
            }
        },
        "gender": {
            "descriptions": {
                "ja": "性別",
                "en": "gender"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "string",
                "enum": [
                    "male",
                    "female",
                    "other",
                    "unknown"
                ],
                "values": [
                    {
                        "value": "male",
                        "descriptions": {
                            "ja": "男性",
                            "en": "male"
                        },
                    },
                    {
                        "value": "female",
                        "descriptions": {
                            "ja": "女性",
                            "en": "female"
                        },
                    },
                    {
                        "value": "other",
                        "descriptions": {
                            "ja": "その他",
                            "en": "other"
                        },
                    },
                    {
                        "value": "unknown",
                        "descriptions": {
                            "ja": "不明",
                            "en": "unknown"
                        }
                    }
                ]
            }
        },
        "active": {
```

```
            "descriptions": {
                "ja": "Patientレコードがアクティブ使用中",
                "en": "Patient record is in active use"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "boolean",
                "values": [
                    {
                        "value": true,
                        "descriptions": {
                            "ja": "アクティブ",
                            "en": "active"
                        },
                    },
                    {
                        "value": false,
                        "descriptions": {
                            "ja": "非アクティブ",
                            "en": "inactive"
                        }
                    }
                ]
            }
        },
        "bmi": {
            "descriptions": {
                "ja": "BMI値",
                "en": "BMI"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "number",
                "unit": "kg/m^2"
            }
        },
        "bodyTemp": {
            "descriptions": {
                "ja": "体温",
                "en": "body temperature"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "number",
                "unit": "Celsius"
            }
        },
        "bodyWeight": {
            "descriptions": {
                "ja": "体重",
                "en": "body weight"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "number",
                "unit": "kg"
```

```
            }
        },
        "bloodPressure": {
            "descriptions": {
                "ja": "血圧",
                "en": "blood pressure"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "array",
                "items": {
                    "type": "number",
                    "unit": "mmHg"
                },
                "minItems": 2,
                "maxItems": 2
            }
        },
        "heartRate": {
            "descriptions": {
                "ja": "心拍数",
                "en": "heart rate"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "number",
                "unit": "/min"
            }
        },
        "pulseOximetry": {
            "descriptions": {
                "ja": "パルスオキシメータ（経皮的酸素飽和度）値",
                "en": "pulse oximetry"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "number",
                "unit": "%"
            }
        }
    }
}
```